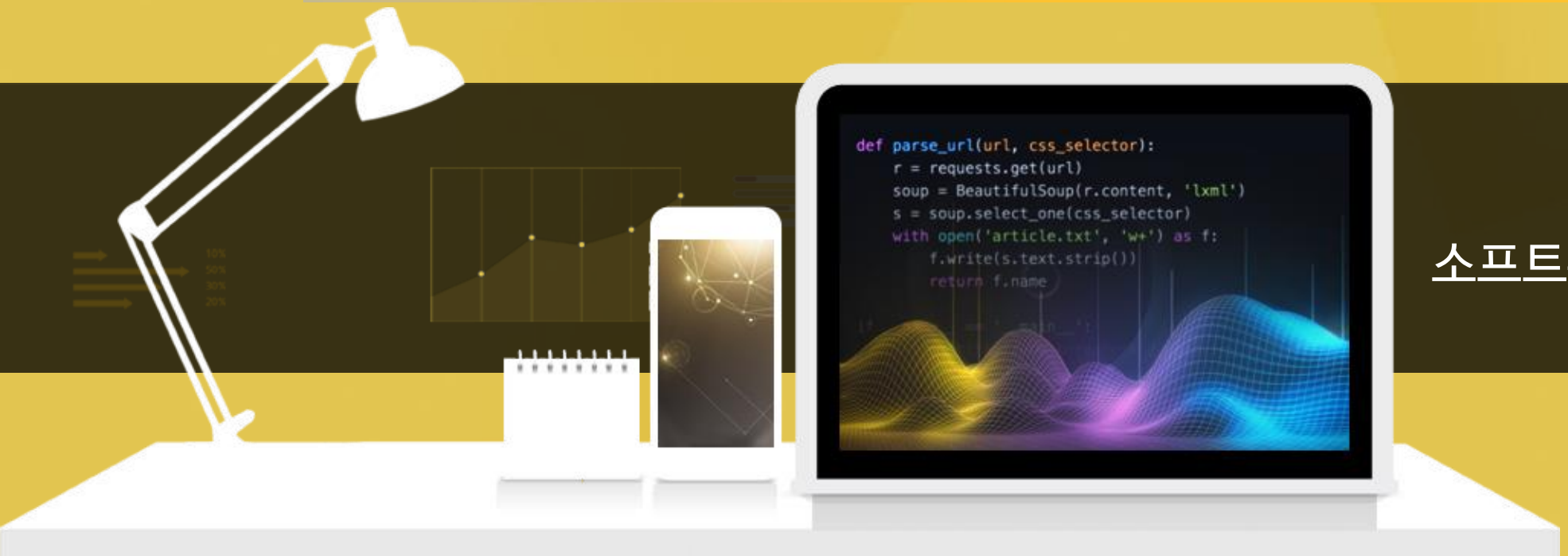
데이터 전처리 실습



소프트웨어융합대학원
진혜진

03 데이터 전처리의 실습

1. 머신러닝 프로세스와 데이터 전처리

- 데이터를 확보한 후 데이터를 정제 및 전처리
- 학습용과 테스트 데이터를 나눠 학습용 데이터로 학습을 실시
- 학습 결과를 평가 지표와 비교하여 하이퍼 매개변수 변환
- 최종적인 모델 생성하여 테스트 데이터셋으로 성능을 측정
- 모델을 시스템에 배치하여 모델을 작동시킴

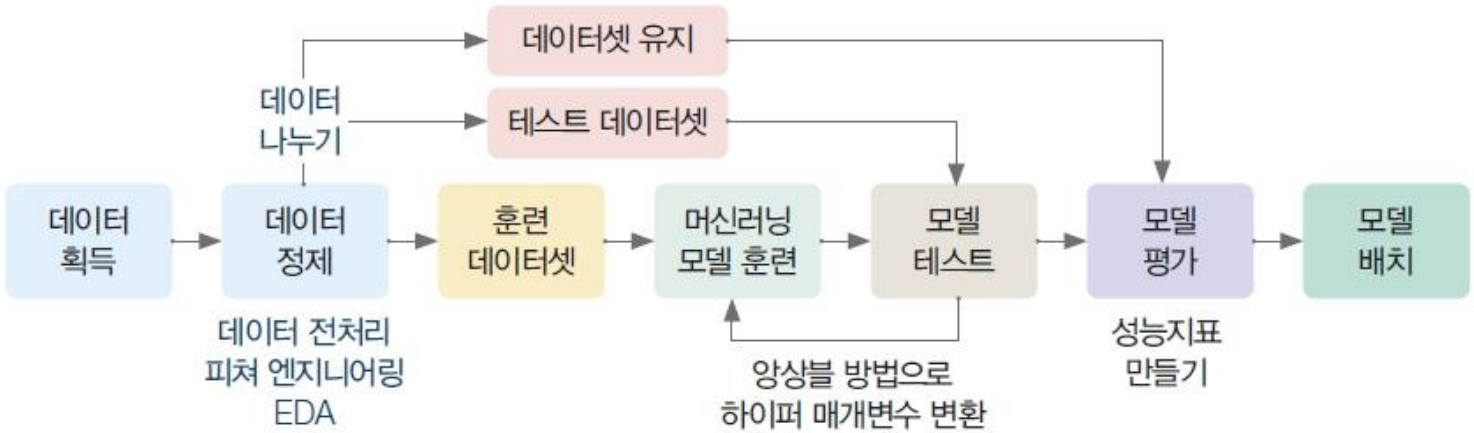


그림 6-3 머신러닝 프로세스

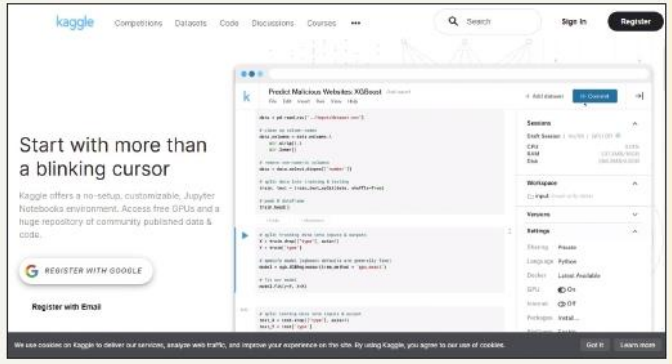
03 데이터 전처리의 실습

데이터 확보를 위한 최적의 장소 : 캐글과 데이콘

- 실제 회사에서 생산 및 처리하고 있는 데이터를 확보한다면 최적이겠지만, 이러한 실제 데이터를 확보하기는 어렵다.
- 다행히 많은 엔지니어들이 연습용 데이터를 제공하고 있다. 대표적으로 캐글(Kaggle)과 데이콘(DACON)이 있다.
- 캐글은 2017년에 구글에 인수되면서 사실상 데이터 분석의 표준적인 프레임워크로 사용되고 있고, 데이콘은 국내 스타트업이 운영하는 데이터 대회 사이트이다.
- 두 사이트 모두 기본적으로 같은 구조를 가지고 있으며 교육과 코드 공유를 통한 데이터 분석 커뮤니티 발전에 도움을 주고 있다.
- 처음 데이터 분석을 배우면서 1차적으로 어느 정도 정리된 캐글과 데이콘의 데이터를 활용하는 것이 좋다.

03 데이터 전처리의 실습

데이터 확보를 위한 최적의 장소 : 캐글과 데이콘



(a) 캐글(Kaggle)의 웹사이트 : https://www.kaggle.com/

(b) 데이콘(DACON)의 웹사이트 : https://dacon.io/

그림 6-5 캐글(Kaggle)과 데이콘(DACON)

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

- 타이타닉 문제는 캐글(Kaggle)에 있는 많은 데이터 중 데이터 분석 입문자가 처음 사용하기 좋은 데이터
- 데이터가 기본적이면서 평가가 쉬움

03 데이터 전처리의 실습

타이타닉 웹페이지

- 개요, 데이터, 코드, 논의, 리더보드, 규칙 등으로 구성
- 처음에는 개요 페이지에서 대회의 내용과 평가지표를 확인
- 평가지표에 맞게 머신러닝 모델링을 실시

- 타이타닉 문제는 배에 타고 있는 승객 대비 살아남을 수 있는 승객을 예측하는 모델로, 'accuracy'라는 지표를 사용

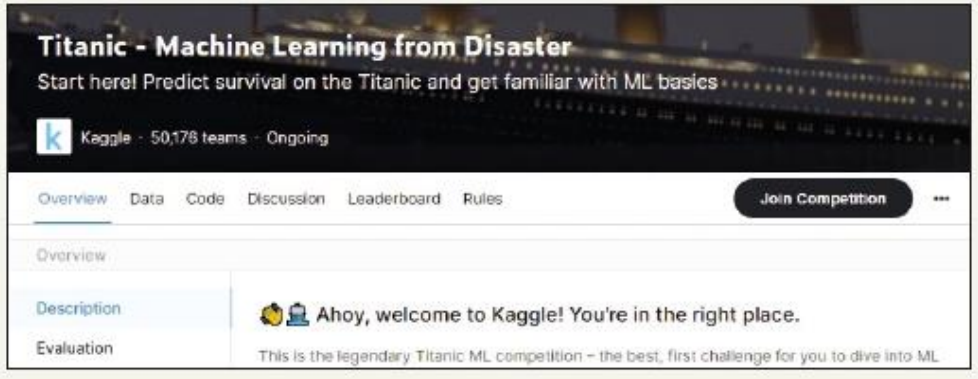


그림 6-6 캐글(Kaggle)의 타이타닉 웹페이지

03 데이터 전처리의 실습

2.1 데이터 확보하기

- https://www.kaggle.com/c/titanic
- [Data] 탭 - 왼쪽 하단 [Download All] 버튼

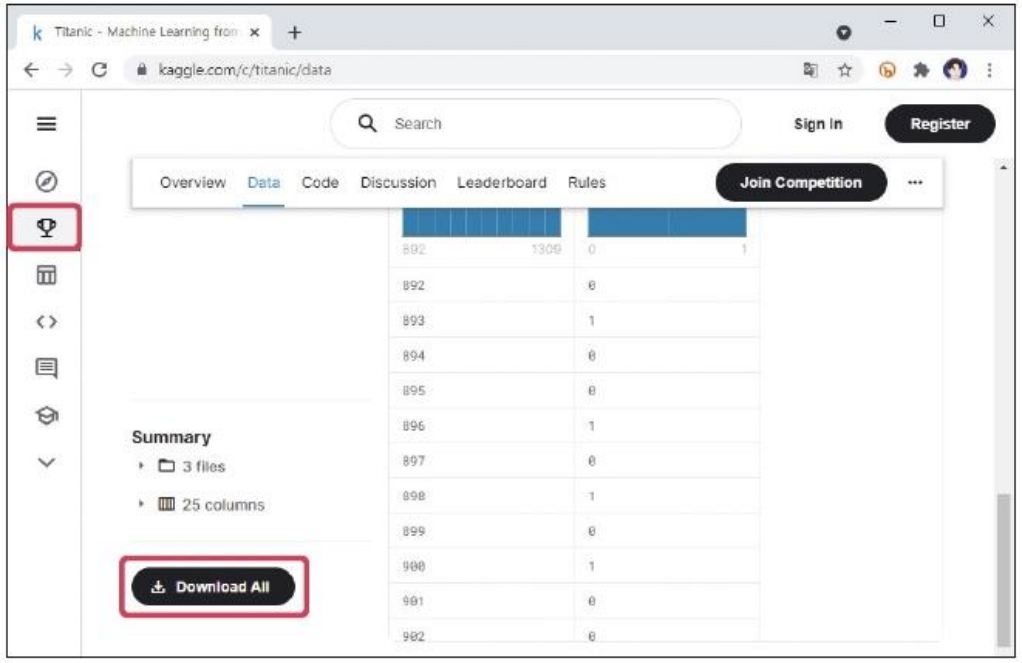


그림 6-7 타이타닉 데이터 다운로드하기

03 데이터 전처리의 실습

- ‘titanic.zip’ 파일의 압축을 풂
  - gender_submission.csv : 데이터 제출 예제 파일로 캐글에 제출하여 평가를 받을 파일의 예시
  - test.csv : 예측되는 탑승객들의 데이터가 있는 파일
  - train.csv : 모델을 학습시키기 위한 데이터가 있는 파일

그림 6-8 titanic.zip 파일에 있는 3개의 데이터 파일

  - ‘train.csv’ 파일 데이터를 사용하여 모델을 만들고
    모델을 ‘test.csv’ 데이터에 적용하여
    결과를 ‘gender_submission.csv’ 파일 형태로 제출

03 데이터 전처리의 실습

- ‘train.csv’ 파일과 달리 ‘test.csv’ 파일에는 y 값,
  즉 탑승객의 생존 유무에 대한 열이 없음
  - 예측에 해당하는 데이터이다

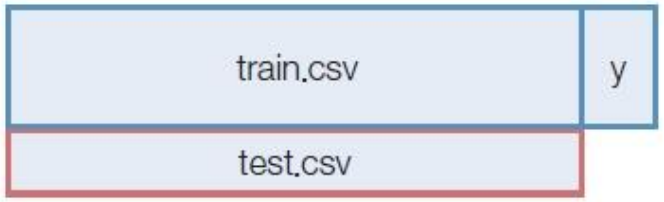


그림 6-9 train.csv와 test.csv의 관계

2.2 데이터 확인하기

- 다운로드한 데이터를 작업 폴더에 넣는다
  - 기본 예제 파일 경로는 ‘C:/source/ch06’
  - 코드 사용이 용이하도록 gender_submission.csv 파일은 삭제

03 데이터 전처리의 실습

In [1]:	import pandas as pd import os import matplotlib.pyplot as plt import numpy as np import seaborn as sns sns.set(style="whitegrid", color_codes=True) DATA_DIR = 'c:/source/ch06/' os.listdir(DATA_DIR)
Out [1]:	['test.csv', 'train.csv']
In [2]:	DATA_DIR = 'c:/source/ch06/' data_files = sorted([os.path.join(DATA_DIR, filename) for filename in os.listdir(DATA_DIR)], reverse=True) data_files
Out [2]:	['c:/source/ch06/train.csv', 'c:/source/ch06/test.csv']

03 데이터 전처리의 실습

In [3]:

```
# (1) 데이터프레임을 각 파일에서 읽어온 후 df_list에 추가
df_list = []
for filename in data_files:
    df_list.append(pd.read_csv(filename))

# (2) 두 개의 데이터프레임을 하나로 통합
df = pd.concat(df_list, sort=False)

# (3) 인덱스 초기화
df = df.reset_index(drop=True)

# (4) 결과 출력
df.head(5)
```

Out [3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

03 데이터 전처리의 실습

2.3 데이터 열 확인하기

표 6-1 타이타닉 데이터셋

변수명	의미	값 종류
Survived	생존 여부	0 = No, 1 = Yes
Pclass	티켓 클래스	1 = 1st, 2 = 2nd, 3 = 3rd
Sex	성별	
Age	나이	
SibSp	타이타닉 밖의 형제자매/부부의 수	
Parch	타이타닉 밖의 부모/자식의 수	
Ticket	티켓 번호	
Fare	티켓 가격	
Cabin	객실번호	
Embarked	승선항구	C = Cherbourg, Q = Queenstown, S = Southampton

03 데이터 전처리의 실습

In [4]:
```
# (1) train.csv 데이터의 수
number_of_train_dataset = df.Survived.notnull().sum()
# (2) test.csv 데이터의 수
number_of_test_dataset = df.Survived.isnull().sum()
# (3) train.csv 데이터의 y 값 추출
y_true = df.pop("Survived")[:number_of_train_dataset]
```

2.4 데이터 노트 작성하기

- 데이터 노트 : 분석해야 하는 데이터에 대한 여러 가지 아이디어를 정리하는 노트
  - 각 데이터의 현재 데이터 타입 올바르게 정의
    - 숫자로 표시되어 있지만 범주형 데이터로 변형이 필요한 경우 등

03 데이터 전처리의 실습

표 6-2 데이터 노트의 예시

변수명	의미	데이터 타입	아이디어
Survived	생존 여부	범주형	Y 데이터
Pclass	티켓 클래스	범주형	
Sex	성별	범주형	
Age	나이	범주형	생존여부에 나이가 영향을 줄까?
SibSp	타이타닉 밖의 형제자매/부부의 수	연속형(int)	
Parch	타이타닉 밖의 부모/자식의 수	연속형(int)	
Ticket	티켓 번호	범주형	
Fare	티켓 가격	연속형(int)	티켓 가격과 pclass와 관련있지 않나?
Cabin	객실번호	범주형	
Embarked	승선항구	범주형	승선항구와 생존률은 영향이 있을까?

03 데이터 전처리의 실습

- 데이터의 모양을 확인할 때 T 함수 사용
  - transpose 함수는 데이터를 가로로 한 줄씩 보여줘 안에 있는 값들을 확인하기 좋음

In [5]:
```
df.head(2).T
```

Out [5]:

	0	1
PassengerId	1	2
Pclass	3	1
Name	Braund, Mr. Owen Harris	Cumings, Mrs. John Bradley (Florence Briggs Th...
Sex	male	female
Age	22.0	38.0
SibSp	1	1
Parch	0	0
Ticket	A/5 21171	PC 17599
Fare	7.25	71.2833
Cabin	NaN	C85
Embarked	S	C

03 데이터 전처리의 실습

2.5 결측치 확인하기

- 열별로 결측치 비율을 확인하여 전략을 세움

In [6]:

```
# (1) 데이터를 소수점 두 번째 자리까지 출력
pd.options.display.float_format = '{:.2f}'.format

# (2) 결측치 값의 합을 데이터의 개수로 나눠 비율로 출력
df.isnull().sum() / len(df) * 100
```

Out [6]:

```
PassengerId    0.00
Pclass         0.00
Name           0.00
Sex            0.00
Age           20.09
SibSp          0.00
Parch          0.00
Ticket         0.00
Fare           0.08
Cabin         77.46
Embarked       0.15
dtype: float64
```

03 데이터 전처리의 실습

- 데이터를 삭제할지 전략적인 의사결정
- 결측치를 채우는 방법을 결정

In [7]:	df[df["Age"].notnull()].groupby(["Sex"])["Age"].mean()
Out [7]:	Sex female 28.69 male 30.59 Name: Age, dtype: float64
In [8]:	df[df["Age"].notnull()].groupby(["Pclass"])["Age"].mean()
Out [8]:	Pclass 1 39.16 2 29.51 3 24.82 Name: Age, dtype: float64

03 데이터 전처리의 실습

In [9]:	df["Age"].fillna(df.groupby("Pclass")["Age"].transform("mean"), inplace=True) df.isnull().sum() / len(df) * 100
Out [9]:	PassengerId 0.00 Pclass 0.00 Name 0.00 Sex 0.00 Age 0.00 SibSp 0.00 Parch 0.00 Ticket 0.00 Fare 0.08 Cabin 77.46 Embarked 0.15 dtype: float64

03 데이터 전처리의 실습

In [10]:	df.loc[61,"Embarked"] = "S" df.loc[829,"Embarked"] = "S"

- 데이터의 특성을 더 잘 나타내는 값으로 채워넣음

2.6 범주형 데이터 처리 : 원핫인코딩

- 데이터 형태에 따라 처리 방법 결정
- df.info() 함수 : 열별로 데이터 타입을 확인
  - 열별로 문자열 리스트 타입으로 정리

03 데이터 전처리의 실습

In [11]:
```
df.info()
```

Out [11]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 11 columns):
 #   Column      Non-Null Count Dtype
---  ------      -------------- -----
0 PassengerId   1309 non-null   int64
1 Pclass        1309 non-null   int64
2 Name          1309 non-null   object
3 Sex           1309 non-null   object
4 Age           1309 non-null   float64
5 SibSp         1309 non-null   int64
6 Parch         1309 non-null   int64
7 Ticket        1309 non-null   object
8 Fare          1308 non-null   float64
9 Cabin          295 non-null   object
10 Embarked     1309 non-null   object
dtypes: float64(2), int64(4), object(5)
memory usage: 112.6+ KB
```

03 데이터 전처리의 실습

- 데이터의 타입을 정리

In [12]:

```
object_columns = ["PassengerId", "Pclass", "Name",
                  "Sex", "Ticket", "Cabin", "Embarked"]
numeric_columns = ["Age", "SibSp", "Parch", "Fare"]

for col_name in object_columns:
    df[col_name] = df[col_name].astype(object)
for col_name in numeric_columns:
    df[col_name] = df[col_name].astype(float)

df["Parch"] = df["Parch"].astype(int)
df["SibSp"] = df["SibSp"].astype(int)
```

03 데이터 전처리의 실습

- 데이터를 원핫인코딩으로 처리

In [13]:
```
def merge_and_get(ldf, rdf, on, how="inner", index=None):
    if index is True:
        return pd.merge(ldf,rdf, how=how,
                        left_index=True, right_index=True)
    else:
        return pd.merge(ldf,rdf, how=how, on=on)
```

In [14]:
```
one_hot_df = merge_and_get(
    df, pd.get_dummies(df["Sex"], prefix="Sex"), on=None, index=True)
one_hot_df = merge_and_get(
    one_hot_df, pd.get_dummies(
            df["Pclass"], prefix="Pclass"), on=None, index=True)
one_hot_df = merge_and_get(
    one_hot_df, pd.get_dummies(
            df["Embarked"], prefix="Embarked"), on=None, index=True)
```

03 데이터 전처리의 실습

2.7 데이터 시각화 진행하기

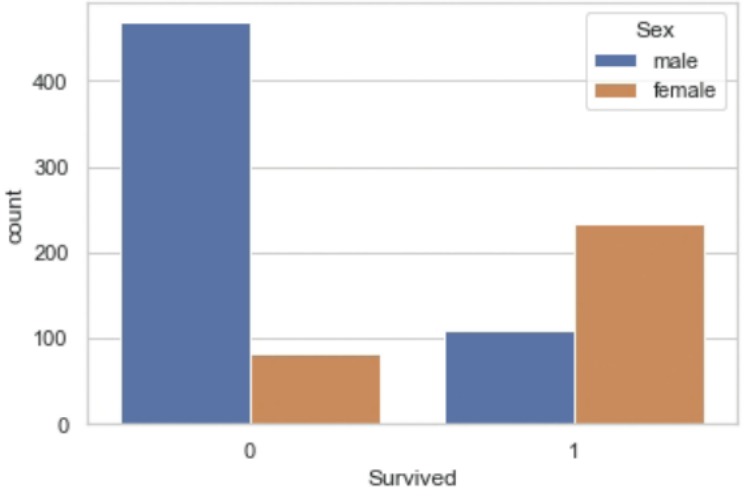
- y 값과 각 범주형 타입 간에 어떤 관계가 있는지를 확인
- 열별로 y_true 데이터와 합쳐서 비교 그래프로 나타내어 각 열이 생존 여부에 영향을 주는지 시각적으로 확인
  - 데이터 유형별로 y_true 데이터의 분포 변화가 있는가

03 데이터 전처리의 실습

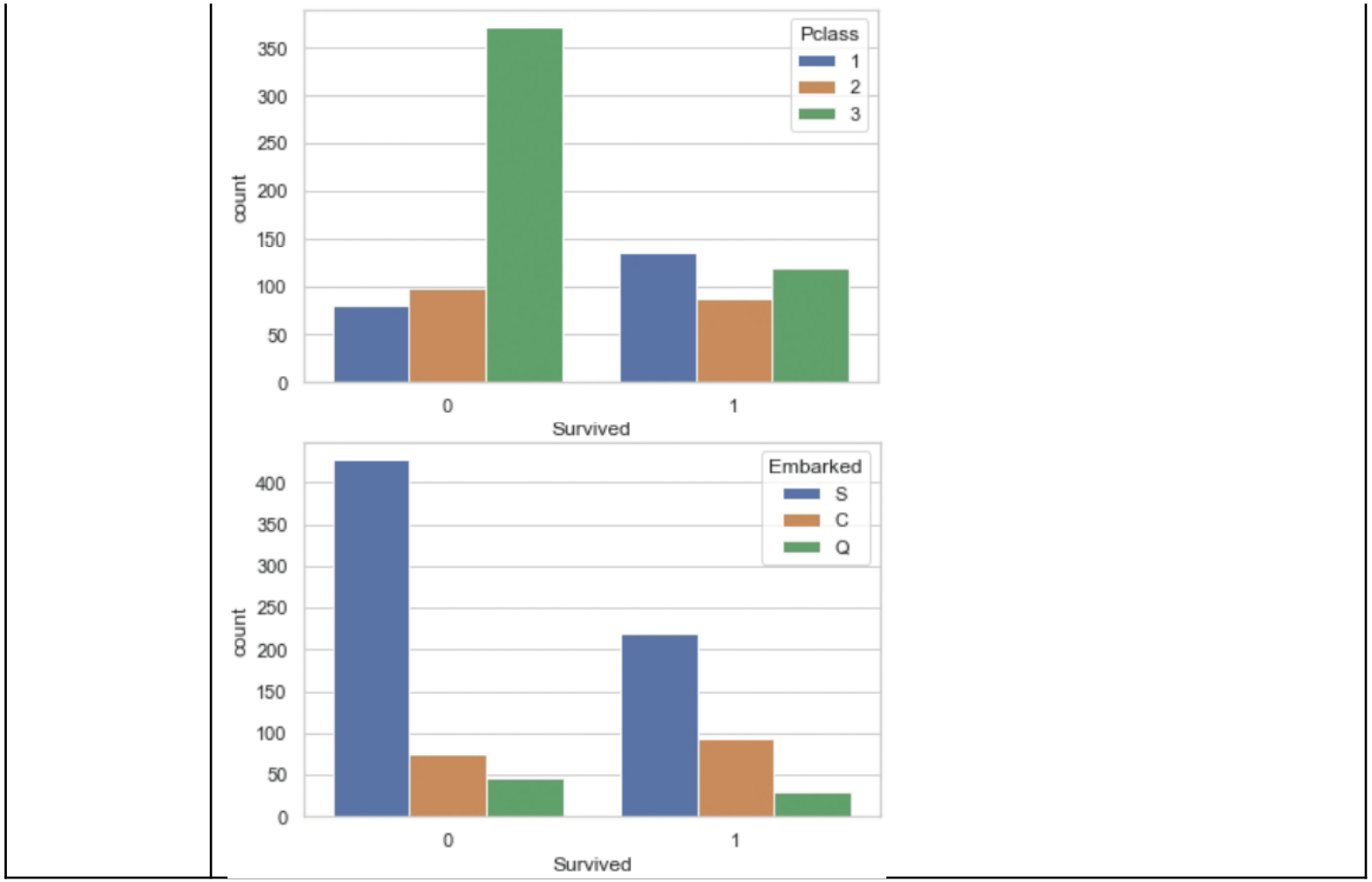
In [15]:

```
temp_columns = ["Sex", "Pclass", "Embarked"]
for col_name in temp_columns:
    temp_df = pd.merge(
            one_hot_df[col_name], y_true, left_index=True, right_index=True)
    sns.countplot(x="Survived", hue=col_name, data=temp_df)
 plt.show()
```

Out [15]:



03 데이터 전처리의 실습



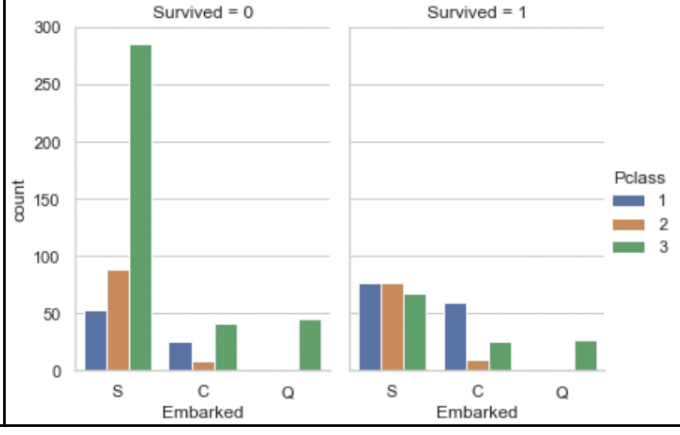
03 데이터 전처리의 실습

- 범주형 데이터 간 상관관계 분석

In [16]:

```
temp_df = pd.merge(one_hot_df[temp_columns],
                   y_true, left_index=True,
                   right_index=True)
g = sns.catplot(x="Embarked",
                hue="Pclass",
                col="Survived",
                data=temp_df,
                kind="count",
                height=4, aspect=.7);
```

Out [16]:

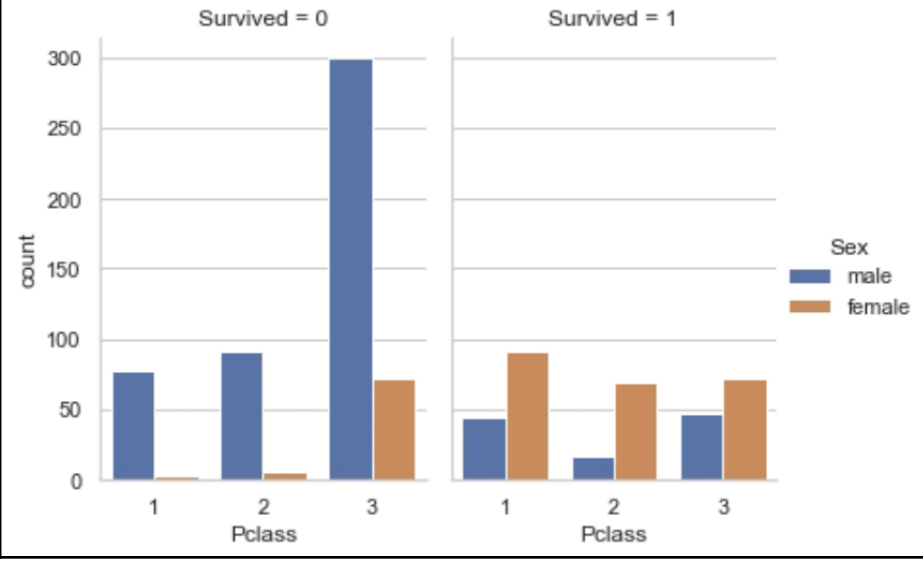


03 데이터 전처리의 실습

In [17]:

```
temp_df = pd.merge(
    one_hot_df[temp_columns],
    y_true, left_index=True,
    right_index=True)
g = sns.catplot(x="Pclass",
                hue="Sex", col="Survived",
                data=temp_df, kind="count",
                height=4, aspect=.7)
```

Out [17]:

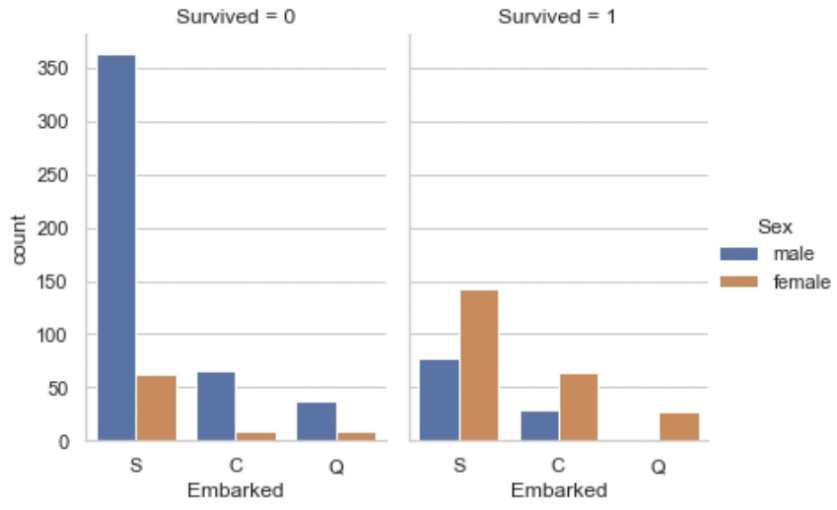


03 데이터 전처리의 실습

In [18]:

```
temp_df = pd.merge(
    one_hot_df[temp_columns],
    y_true, left_index=True,
    right_index=True)
g = sns.catplot(
    x="Embarked", hue="Sex",
    col="Survived",
    data=temp_df, kind="count",
    height=4, aspect=.7);
```

Out [18]:



03 데이터 전처리의 실습

- Heatmap 함수 : 상관계수(correlation) 데이터로 확인
  - corr 함수로 상관계수 계산

In [19]:

```
crosscheck_columns = [col_name for col_name in
one_hot_df.columns.tolist()
    if col_name.split("_")[0] in temp_columns and "_" in col_name ] +
["Sex"]

# temp 열
temp_df = pd.merge(one_hot_df[crosscheck_columns],
    y_true, left_index=True, right_index=True)

corr = temp_df.corr()
sns.set()
ax = sns.heatmap(corr, annot=True, linewidths=.5, cmap="YlGnBu")
```

03 데이터 전처리의 실습

Out [19]: